REMARKS/ARGUMENTS

This Amendment is in response to the Office Action dated October 5, 2007, and is being filed with a one month extension of time. Claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 are pending. No claims have been amended or canceled.

Claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28 and 31-33 are rejected under 35 USC 103(a) as being unpatentable over Sharma et al (US 2003/0204645; hereinafter Sharma) in view of Chiang et al. (US 6,948,174; hereinafter Chiang) and Fry (US 6,880,125). Applicant respectfully disagrees.

The present invention addresses the situation where multiple client applications need to access respective data sources that are stored in different formats and are not directly accessible by the other client applications. Accessibility is provided by storing data from the respective data sources in a database at a server in a manner where the data is standardized. The invention provides this standardized data by providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, and modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files. Each of the XML files is then submitted to an import repository at the server. Each of the XML files in the import repository are validated against a document type definition (DTD) corresponding to the respective data sources. The validated XML files are parsed, and name/value pairs are stored in the database at the server according to a hierarchy specified by the corresponding DTD. Thus, the XML files are collected and

validated in the import repository before being stored in the database.

The collecting of the XML files in the import repository provides the following advantages: 1) it provides a separate collection of XML files to ensure that data sources can be completely validated and recorded before entering the database; 2) a separate staging area keeps the XML files in isolation of the operational database, and hence minimizes the impact on the integrated data warehouse (if things go wrong); and a separate staging area for a client applications provides an ability to track transactions independent of the database. (See specification at p. 17, lines 14-23) Because the XML files are provided and validated in the import repository in the claimed manner, prior to being stored in the database, true standardization of the data from the data sources is provided.

In contrast, Sharma is directed to providing a servlet container-based Web service endpoint in a remote procedure call-based distributed computing system [003]. Sharma describes a computing system which provides a service associated with a service endpoint class that implements a service endpoint interface by packaging the service endpoint class and interface in an archive file. The computing system may define a service endpoint based on information included in the archive file and modify the archive file with information associated with the service endpoint definition. The modified archive file may then be deployed on a servlet container operating in the computing system. [0011]. The computing system may create a Web Services Description Language (WSDL) document that describes the service endpoint based on the information contained in the modified archive file and export the WSDL document such that a remote computing system may use the WSDL document to access the service endpoint. [0012].

As is known in the art, a remote procedure call (RPC) is a technology that allows a computer program to cause a subroutine or procedure to execute in another address

space, typically on another computer on a shared network, without the programmer explicitly coding the details for this remote interaction. That is, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote. RPC Web services present a distributed function (or method) call interface that is sometimes implemented by mapping services directly to language-specific functions or method calls (Wikipedia).

By Contrast, the present invention has nothing whatsoever to do with RPC, whereby the client is invoking procedures or methods on the server. Instead, the claims of the present of invention are directed to, inter alia, client side adapter APIs.

Sharma Fails to Teach or Suggest the Elements of Claim 31

Applicant agrees with the Examiner that Sharma fails to specifically disclose saving XML format data from respective data sources in XML files, as partially recited in step (b) of claim 31, and fails to specifically disclose "validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources," as recited in step (d) of claim 31. However, applicant would go further, and submits that Sharma fails to teach or suggest the remaining elements of claim 31 as well.

First, Sharma fails to teach or suggest "providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format," as recited in step (a).

The Examiner cited FIG. 5, paragraphs 5, 8-10, 37, 115, and 127 of Sharma for teaching this step. However, paragraphs 5 and 6 merely describe a generic client/server computing environment and provide the definition of remote procedure calls (RPCs), while paragraphs 8-10 and 37 discuss XML and XML based RPC services and environments. Paragraph 127 describes that the APIs may be implemented to support an extensible type

mapping between XML data types and Java types. Although paragraph 115 describes that a client may use client-side APIs, which presumably the Examiner considers analogous to the claimed "adapter API", Sharma's client side API's use a stub 515 to invoke a remote method on service endpoint 555 *in a server*. Sharma does not teach or suggest in this passage, or any other that Applicant could find, where client side API's "translate data in the respective data sources into XML format," as claimed.

Moreover, Applicant could find no teaching or suggestion in Sharma of a plurality of client applications, "wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications," as further recited in step (a). Sharma may teach that his system may include a plurality of clients [065], and that the clients may require data stored another computing node, typically a server [005], but Sharma falls short in teaching that the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications.

Because Sharma fails to teach or suggest "an adapter API at each of the client applications that provides a first set of methods...," in step (a) as argued above, it follows that Sharma cannot teach or suggest "modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition," as recited in step (b).

The Examiner cited paragraphs 7, 72, 91, 115, 127, and 130 of Sharma for teaching this step. Paragraph 115 may describe that the client side APIs uses stub 515 to invoke a remote method on service endpoint 555. However, the remote method invoked is on the server, rather than a method of the client API. In addition, the remote method invoked in Sharma fails to "convert data in respective data sources into XML format according to a

registered schema definition," as claimed.

Sharma also fails to teach or suggest "submitting each of the XML files to an *import repository* at a server", as recited in step (c), in combination with "parsing the validated XML files in the import repository and storing name/value pairs in *a database* at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications," as recited in step (e).

The Examiner cites paragraphs 5, 115 and 127 for submitting the XML files to an import registry at the server. However, none of these passages describe a process by which XML files created by client side APIs are submitted to an import registry at the server. Sharma also fails to teach a database of the server that is separate from the import registry. In fact, an electronic search of Sharma for the terms "import registry" and "database" revealed no matches. Because Sharma fails to teach a database apart from an import registry, Sharma cannot teach or suggest "parsing the validated XML files in the import repository and storing name/value pairs in *a database* at the server according to a hierarchy specified by the corresponding DTD."

Independent claims 32 and 33 include the same or similar limitations as claim 31. Therefore, claims 32 and 33 are allowable for at least the same reasons as claim 31.

Secondary References Fail to Cure Sharma

As stated above, the Examiner admits that Sharma fails to specifically disclose saving XML format data from the respective data sources in XML files. The Examiner cited Chiang for teaching saving the XML format data an XML file.

Chiang discloses using the XML file to convert languages between a user application and a server. Although Chiang discloses the use of an XML file, Chiang is concerned with a user application and a server being able to communicate, not the

standardization of data from multiple data sources in different formats at a database. Chiang does not teach or suggest how to collect, validate, parse, or store XML files from multiple data sources in different formats, such that the standardization of data from the data sources is provided across user applications. Even assuming that multiple user applications can communicate with the server as disclosed by Chiang, this only teaches how each individual user application can communicate with the same server. This is not a teaching of how the data from these multiple user applications are truly standardized for access by each other.

Sharma in view of Chiang thus fails to teach or suggest modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files, submitting each of the XML files to an import repository at a server, validating each of the XML files in the import repository against a DTD corresponding to the respective data sources, and parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources, as recited in amended independent claims 31, 32, and 33.

The Examiner cited Fry for teaching validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources. However, Fry likewise fails to provide any teaching regarding the standardization of data from multiple data sources in different formats via client side adapter APIs and the claimed process. Therefore, a combination of Sharma, Chiang, and Fry also fails to teach or suggest independent claims 31, 32, and 33.

In view of the foregoing, it is submitted that claims 31-33 are allowable over the cited

references. Because the secondary references stand or fall with the primary references, dependent claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28 are allowable because they are dependent upon the allowable independent claims. Accordingly, Applicant respectfully requests reconsideration and passage to issue of claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 as now presented.

Applicants' attorney believes this application in condition for allowance. Should any unresolved issues remain, Examiner is invited to call Applicants' attorney at the telephone number indicated below.

Respectfully submitted,

/Stephen G. Sullivan/
Stephen G. Sullivan
Attorney/Agent for Applicant(s)
Reg. No. 38329
Telephone No.: 650 969-7474

Date: January 9, 2008